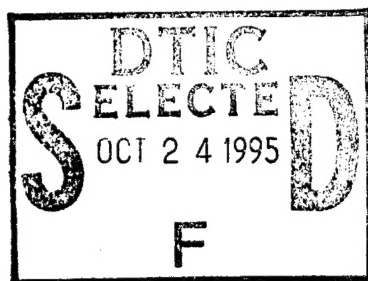


NAVAL POSTGRADUATE SCHOOL

Monterey, California



19951023 173

THESIS

SOFTWARE REUSE AND ITS INHERENT LEGAL IMPEDIMENTS

by

Claxton R. Johnson, Jr.

June 1995

Principal Advisor:
Associate Advisor:

William Short
Mark Stone

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 5

REPORT DOCUMENTATION PAGEForm Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE June 1995	3. REPORT TYPE Master's Thesis	
4. TITLE AND SUBTITLE SOFTWARE REUSE AND ITS INHERENT LEGAL IMPEDIMENTS.		5. FUNDING NUMBERS	
6. AUTHOR(S) Claxton R. Johnson, Jr.			
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the US Government.			
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) Software reuse has become a critical issue as public and private entities struggle to control cost in the expensive field of applications development and authors of software products act to protect their intellectual property rights. Software customers who are engaged in end-use development hope to be able to negotiate reuse contracts that allow them to economically develop complete applications. This thesis investigates the history and discusses the legal issues surrounding the reuse of software as intellectual property. It concludes by discussing several measures for dealing with the legal barriers to software reuse.			
14. SUBJECT TERMS CARDS, FAR, DFARS, Software Reuse, Liability		16. PRICE CODE	
		15. NUMBER OF PAGES 59	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL
NSN 7540-01-280-5500		Standard Form 298 (Rev. 2-89)	

Approved for public release; distribution is unlimited.

SOFTWARE REUSE AND ITS INHERENT LEGAL IMPEDIMENTS

Claxton R. Johnson, Jr.
Captain, United States Marine Corps
B.S., Hampton University, 1987


Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
June 1995

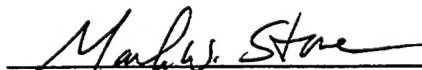
Author:


Claxton R. Johnson, Jr.

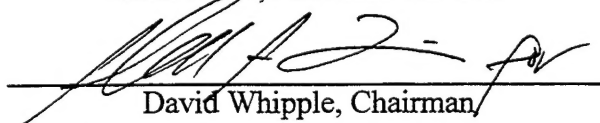
Approved by:



William Short, Principal Advisor



Mark Stone, Associate Advisor



David Whipple, Chairman
Department of Systems Management

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

ABSTRACT

Software reuse has become a critical issue as public and private entities struggle to control cost in the expensive field of applications development and authors of software products act to protect their intellectual property rights. Software customers who are engaged in end-use development hope to be able to negotiate reuse contracts that allow them to economically develop complete applications. This thesis investigates the history and discusses the legal issues surrounding the reuse of software as intellectual property. It concludes by discussing several measures for dealing with the legal barriers to software reuse.

TABLE OF CONTENTS

I. INTRODUCTION	1
A. PURPOSE	1
B. GENERAL SOFTWARE DEVELOPMENT ISSUES	1
C. RESEARCH OBJECTIVES	3
1. Investigative Questions	3
2. Scope	3
D. ORGANIZATION	4
II. BACKGROUND	5
A. SOFTWARE HISTORY	5
B. SOFTWARE REUSE	6
1. Definition and Use	6
2. Background	8
C. SOFTWARE REUSE: LEGAL HINDRANCES	10
III. SOFTWARE REUSE LEGAL ISSUES	13
A. INTRODUCTION	13
B. ASSUMPTIONS AND DEFINITIONS	13
C. LIABILITY	15
1. Liability on the basis of a contract	16
2. Liability on the basis of tort	16
3. Statutory Liability	17
D. DATA RIGHTS ISSUES	18
1. Copyright Law	20
2. Patent Law	22
E. THE EFFECTS OF FEDERAL REGULATIONS	22

1. FAR vs. DFARS	23
IV. METHODOLOGY AND ANALYSIS	28
A. METHODOLOGY	28
1. Methods.....	28
2. Justification of Methodology Choice.....	29
3. Methodology Perspectives	29
4. Data Analysis Methodology	30
B. ANALYSIS	30
C. RESPONSES FROM PHONE CALLS	31
V. CONCLUSION AND RECOMMENDATIONS.....	38
A. THE MAJOR CONCERNS	38
B. INHERENT PROBLEMS.....	39
C. RECOMMENDATIONS	40
1. Education	40
2. Acquisition Professional Recommendations	41
D. AREAS FOR FURTHER RESEARCH.....	43
LIST OF REFERENCES	46
INITIAL DISTRIBUTION LIST.....	48

I. INTRODUCTION

A. PURPOSE

The main purpose of this thesis is to address the concerns in the software reuse community about perceived legal impediments in using and developing reusable components. This thesis is intended for use by managers who are responsible for formulating or implementing plans of operations for Software Reuse. It is intended to give insight into some of the problem elements which should be taken into account when using software reuse as an acquisition requirement. It is not an opinion of law, but is intended to promote discussion and serve as the occasion for taking specific questions to legal counsel for their opinion. By publicizing this thesis to policy makers and implementers, the researcher hopes to initiate positive changes in the software reuse, software acquisition, and legal communities.

B. GENERAL SOFTWARE DEVELOPMENT ISSUES

The software costs of Department of Defense (DoD) systems continue to rise due to the need for larger and more complex systems and the increasing percentage of key problems resolved through software development. Existing software development methods are ill-equipped to deal well with the increased size and complexity of these systems. To reduce software costs, there is a need for methods which address the chronic problems faced by the developers of these systems. These problems include:

- Inefficient use of experienced engineers.
- Inaccurate estimates of development schedule and costs.
- Vague and incomplete requirements.

Corporate knowledge is seldom shared among Government contractors even when software requirements are similar. As a result, experience from previous engineers is often lost.

Inaccurate estimates of a project's cost and development schedule result from the incomplete understanding of system complexities at bid and proposal time and from the lack of accurate and consistent software metrics from past projects. Inaccurate estimates mean that projects are at increased risk of failing to meet their schedule and budget. In an increasingly competitive environment which dictates very aggressive bids and schedules, there is little room for this type of error.

Because customers order "systems", Statements of Work often do not address specific software requirements. The job of sorting out software requirements is then left to system engineers who may not adequately address software impacts and issues. If software requirements are vaguely specified, they will be clarified during design and implementation, but not necessarily in a way that is consistent with the customer's needs. If software requirements are incomplete, omissions may not be discovered until after delivery. Either way, the result often leads to a dissatisfied customer.

In recent years much attention has been paid to software reuse because it is recognized as a key means for reducing some of the problems associated with software development mentioned earlier and obtaining higher productivity in the development of new software systems.. Through reuse, prior expertise from those who originally wrote the code is incorporated into the whole program, reducing error content thereby taking advantage of the efforts of experienced engineers. The primary economic benefit of software reuse is cost reduction. The cost of that portion of reused code includes only the cost to purchase it from the reuse library with very little development cost. Reuse of an existent software object generally costs much less than creating a new one and as a result schedule and costs are more accurately estimated. The reduction in development time allows engineers more time to validate the requirements thereby reducing one of the inherent problems in software development which is vague and incomplete requirements. In addition, software reuse has provided the technical benefit of reduced error content leading to higher quality. [Ref. 1:50]

In DoD, however, many perceive that legalities are prohibiting widespread software reuse. Although legal issues can impact the management of software reuse, if treated within the context of a business strategy at the onset of a reuse program, risks and liability can be reduced. The purpose of this thesis is to provide insight on what and how to inhibit the legalities which do prohibit software reuse.

C. RESEARCH OBJECTIVES

The research objectives of this thesis are to identify what the inherent legal acquisition problems prohibiting software reuse are, and what can be done to minimize them. Electronic mail queries were transmitted and phone calls were made to assist in attaining the research objectives. The respondents came from the Chief of Staff of the Army, Intellectual Property Law Division; the Defense Information Systems Agency (DISA), Software Development Section; and the Marine Corps Tactical Support Systems Activity (MCTSSA).

1. Investigative Questions

Investigative questions which will be addressed are:

- a. What are the legal issues involved in reusing software?
- b. How can the acquisition process address software reuse?
- c. What acquisition problems are inherent in software reuse?
- d. What should the acquisition professional be aware of when required to include software reuse initiatives in a contract?

2. Scope

This thesis is limited to implementing the software reuse concept only as it pertains to acquisition issues. The thesis will address the following:

- a. Identification of existing solutions to controlling some of the legal problems associated with reusing software.

- b. Identification of some of the acquisition procedures for implementing and using the software reuse concept.
- c. Identifying some of the problems associated with software reuse as they pertain to the intellectual property rights.

D. ORGANIZATION

Chapter II defines software reuse and gives background information on how it has evolved into its present uses. Chapter III identifies the main intellectual property concerns which affect software reuse. Chapter IV explains the methodology used to gather feedback and to aid in the analysis of the research. In Chapter V, conclusions are drawn from the research and recommendations to correct the barriers are presented.

II. BACKGROUND

A. SOFTWARE HISTORY

Software upgrading has become increasingly important in this time of rapid evolution of Threat Weapon Systems computer technology. Software has become so prominent in U.S. weapon systems design that it has proven to be an influencing factor on overall system design about 50 percent of the time since the early 1970's. [Ref. 2:2] However, software as a major component of computerized systems is a relatively new development. It was not until the late 1960's that hardware and software components of digital systems began to progress along separate paths.

Hardware development has been revolutionary, producing faster and ever more advanced and capable machines. This revolution has been brought about by the marriage of the silicon chip and electrical engineering, and has resulted in such things as Very High Speed Integrated Circuits (VHSIC). As a result, computers have exponentially increased their capability. This revolutionary process has gone from 16 bit architecture utilizing 60,000 word memories to 32 bit architecture utilizing 5.0+ million word memories. [Ref. 3:2] Paralleling the hardware revolution has been a software evolution. Although a slow process often being equated to a "black art," the evolution of software has produced exponential results in such a short period. [Ref. 4:31] In approximately forty years since the introduction of the first digital system, software evolution has produced hundreds of languages spanning four levels of complexity¹, multitudes of design and instruction set architectures, and a plethora of development techniques and styles.

Webster's dictionary defines software as "the entire set of programs, procedures, and related documentation associated with a system and especially a computer system". Software is at best conceptual, lending itself to the nature of an art more than science or

¹ Levels of complexity for computer languages are categorized into four groups: machine languages, assembly languages, higher order languages, and application generators. [Ref. 4] Each language meets a specific need at a particular level of programming

engineering. Yet, it is one of the most critical resources of the Department of Defense for the production of today's sophisticated, high-technology weapon systems. The DoD considers weapon systems software to be on the "critical path" of systems development. [Ref. 2:1]

B. SOFTWARE REUSE

The notion of software reuse is not new. The concept of treating software elements as black box components and composing systems based on well-understood architectures which use those components was first proposed almost twenty-seven years ago. The concept of software reuse was first formally introduced in 1968 when it was foreseen as a subset of software development: a segment of the industry that would design, code, test and distribute reusable software components, much the same as hardware components, such as computer chips and resistors [Ref. 1:481]. It is only recently, however, that significant progress is being made in this area. Since 1983, companies have begun discovering that a reuse-based business strategy improves their competitive and profit base. [Ref. 5:1] There are a couple of reasons for this trend. One reason being the reduction in the defense budget which resulted in a major push for initiatives which reduce costs. Another reason is the continuing increase in the cost of computer software which has overtaken the growth rate of hardware. The cost of software is expected to continue to rise at phenomenal rates. [Ref. 6:1]

1. Definition and Use

"Reuse may be defined as developing a new software system using existing software components probably with some new components as well". The object of reuse is to avoid building a new software product from scratch. [Ref. 7] Additional amplification of the definition of software reuse is that it is the application of a reusable component to more than one application system. Reuse may occur within a system (e.g., F-14A), across similar systems (e.g., M1, Bradley), or in widely different systems (e.g., user interface services). [Ref. 5]

The Toshiba Corporation has had an active software factory since 1976. The software written there is for real-time, highly critical functions, such as nuclear power stations, electric utility networks, chemical process plants, and steel rolling mills. One of the concepts embodied at the factory is reusability. An article written in 1981 stated software development was increasing at the rate of 18% per year, but due to profitability constraints, the corporation could not afford to hire additional workers. Reusable software made up the difference. [Ref. 1:11]

The secret to the Toshiba Corporation's success was through methodologies that are currently available. The environment for development was consistent over the entire time period; one computer with a standardized tool set. Code was targeted for use in one of four mini or midi-computers, and more than five types of microcomputers. Languages used were PL/7, FORTRAN, assembler, and TPL, which is a high level language for real-time industrial control minicomputers. Cross-assemblers were used to convert the developed code into object code for the targeted system. Registered programs were stored in a library under two categories, standard programs and job oriented programs, each of which could be accessed by entering a program name or function code from developers terminals. [Ref. 1:12] In this way software reuse became an integral part of their software development process.

Like the Toshiba Corporation, the Department of Defense also believes that reuse principles, when integrated into its acquisition practices and software engineering process, can provide a basis for dramatic improvement in the way software-intensive systems are developed and maintained over their life-cycle. Therefore, the DoD has elected to accelerate the incorporation of software reuse into its business strategy. The reasons for doing so are to increase productivity, improve the quality and reliability of software-intensive systems, enhance system interoperability, identify and manage technical risk, and shorten development and maintenance time. [Ref. 8:1]

There is common belief among software reuse proponents that software reuse will eventually happen whether the DoD takes an active role or not. The challenge is to

position the DoD to accelerate its use and to reap its benefits. The Department must make a careful and deliberate analysis of the business, technical, and developmental context in which a DoD software reuse strategy will be implemented.

2. Background

DoD must combat rising software development costs. Some 1995 estimates put DoD's software costs as high as \$42 billion. The commercial sector has shown that reuse can reduce overall software expenses by between 10 percent and 15 percent and cut maintenance costs by 20 percent. [Ref. 7:1]

Recent events such as the DoD Software Master Plan and the new Advanced Research Projects Agency (ARPA) initiatives, have indicated a renewed interest within DoD to implement an effective software reuse program. Although this goal was attempted previously, it met with poor results. Two reasons why reuse failed were poor management and failure to forecast legal inhibitors to reuse.

Management was poor because of poor organizational structure. Last year, in a report accompanying the fiscal 1993 Defense appropriations bill, the House Appropriations Committee said:

The department, with its decentralized approach, runs the risk of permitting the many organizations participating in reuse initiatives to misdirect or duplicate reuse efforts. [Ref. 8:75]

Three years after the Defense Department established its software reuse program, DoD officials are restructuring it and putting a single office in charge. The result will be that no longer will software reuse be a voluntary type program, but rather all DoD components will have a role and accompanying responsibilities and will participate in the Software Reuse Initiative (SRI). The Software Reuse Initiative came about almost solely as a way for the Department of Defense to reduce cost in these days of reduced budgets. The first SRI efforts will be to develop an SRI strategic plan that includes specific objectives and projects.

Currently, DoD officials have set up a new reuse organizational structure. The SRI program will be the responsibility of Emmitt Paige Jr., Assistant Secretary of Defense for Command, Control, Communications and Intelligence (ASDC3I). Paige is responsible for coordinating SRI efforts with the UnderSecretary of Defense for Acquisition and Technology, Director of Defense Research and Engineering (DDR&E), and Advanced Research Projects Agency. [Ref. 8:73] The original reuse program management responsibility was divided between the DDR&E and the former Director of Defense Information, Mr. Paul Strassmen. This new SRI policy of centralizing software reuse under the ASDC3I yields the primary benefit that only one organization will be responsible for promulgating reuse policy within DoD and providing program direction and oversight which has been lax in the past. This office will also be responsible for ensuring the SRI program achieves its goals and ensuring budgets are used properly.

Under the SRI program a new software reuse office has been established within the Defense Information Systems Agency's (DISA) Center for Information Management. This office acts as the primary DoD reuse organization responsible for developing an overall reuse plan, establishing reuse procedures and defining all reporting requirements. The current acting director of this office is Colonel Carl McIntyre. The program office is responsible for overseeing implementation of SRI, however the Services and DoD agencies themselves will be responsible for carrying out the reuse programs. This procedure minimizes the inherent nature of a centralized decision authority to stifle creativity, especially within the Defense Technology fields. This type of structure will also provide the centralized control and management that congressional leaders found lacking in the original reuse program.

Like all new defense technology programs, the SRI also has to be aligned with overall Defense Technology policy. This means having established goals which are in compliance with the current DoD goals concerning technology development. Through the initiative, DoD has established reuse as an essential element of systems life cycle management. The SRI has three major mission goals:

- Bring reuse technology into the DoD mainstream
- Develop an infrastructure that reduces the cost and risk of adopting and sustaining reuse programs.
- Encourage, reward and institutionalize effective software reuse.

The second reason for the early failure of software reuse in DoD was the legal barriers inhibiting reuse, specifically the handling of intellectual property rights.

C. SOFTWARE REUSE: LEGAL HINDRANCES

Since the DoD began investigating strategies for incorporating reuse into systems acquisition and development, various groups have been examining the technical and business aspects of software reuse. Issues such as acquisition regulations and policies, library mechanisms and interoperability have been addressed. However, there has been limited progress within the legal discipline in formulating guidance for Government managers in software reuse in general, and in establishing and operating reuse libraries in particular. [Ref. 6:2] The Central Archive for Reusable Defense Software (CARDS) Program, sponsored by the U.S. Air Force (ESC/ENS), Hanscom AFB, MA., has also been investigating business areas that impact software reuse. As a result, the CARDS Program established an on-going forum, made up of DoD lawyers and contracting officers familiar with software reuse, to examine legal issues and concerns related to software reuse. [Ref. 8:1]

The forum determined legal issues pertaining to copyrights, policies and regulations, and liability as it relates to warranty (or lack of it) need to be addressed. However, the CARDS legal forums have focused on providing guidance to Government managers in establishing and operating reuse libraries. Since Government software reuse libraries have not existed for very long, many library managers are finding it difficult to develop agreements and determine the best legal approach to operations and interactions with others. In general, Government lawyers are not knowledgeable about software reuse

and in turn are not able to provide the necessary support to Government managers. [Ref. 8:1]

These issues pertaining to the legal aspects of software reuse are the primary focus of the next chapter.

III. SOFTWARE REUSE LEGAL ISSUES

A. INTRODUCTION

Since 1986, a policy has existed within the Government to encourage reuse among Government contractors. The policy favors allowing contractors to commercially market software developed under Government contracts. [Ref. 10]

Research in acquisition since 1986 has shown that additional development in the legal and contractual areas is needed to increase the effectiveness of reuse activities. The general conclusion is that software reuse is still not addressed effectively in Request for Proposals (RFPs), contractor proposals, or contracts. Reuse is now "disincentivized" by:

- Complexity of the legal and ownership issues
- Limitations of present Federal Acquisition Regulation (FAR) and Defense Federal Acquisition Regulation Supplement (DFARS) and of proposed changes, with regard to computer software
- Many stakeholders whose needs can enable or prevent successful reuse
- Current lack of adequate incentives to address those needs

B. ASSUMPTIONS AND DEFINITIONS

The following are assumptions which are made throughout this thesis:

- The software reuse library/archive is Government Owned-Government Operated (GOGO) or Government Owned-Contractor Operated (GOCO). This assumption significantly limits exposure because of the concept of the "law of sovereign immunity", which essentially provides for no Government liability except in cases where statutes otherwise provide.
- The library/archive is intended for Government purpose use only. [Note: Libraries that distribute components to subscribers for the purpose of commercial software development are not included in this thesis.]
- Subscribers and suppliers are Government personnel, Government contractors or prospective Government contractors.
- The library/archive contains Government Off The Shelf (GOTS), Commercial Off The Shelf (COTS), and public domain components. For

COTS components, the library supplies the name, address and telephone number of the owner, with a description of the component.

- The library/archive does not negotiate or determine terms and conditions of use with respect to third party users (in the event of redistribution). The library/archive does not act as either dealer/reseller or as an agent for a supplier or subscriber because its liability would rise dramatically.² The subscriber must contact the supplier directly for purchasing, licensing and /or maintenance agreements.
- The library/archive contains no classified components. Government agencies charged with setting standards for computer systems which process classified material have identified no system accessible by modem over commercial telephone lines whose safeguards are adequate for the protection of classified Government materials.

Table 3-1 shows operational definitions prepared by the Joint Integrated Avionics Working Group (JIAWG) Software Task Group. Definitions are adapted from JIAWG (1990, item 1.2). These operational definitions may not correspond precisely with legal definitions.

² The role of dealer/reseller and agent may be inappropriate for other reasons, such as Department policy against the Government actively competing in commercial arenas (see DoDI 7230.7, General Policy).

Reusable Software Term	JIAWG Definition
Reusable Software Object (RSO)	Life-cycle objects that are created during the software development process, and have the potential for reuse. The objects may include (but are not limited to): requirement specifications, design documents (both top level and detailed), source and object code, test specifications, test code, test support data, user manuals, programmer notes, and algorithms. These objects may be textual, graphical, or both; they are usually stored on electronic media; and, in DOD-STD 2167A and in DFARS Parts 270 and 272, they are defined as computer software and computer software documentation.
Developer	A contractor who creates and makes use of RSOs, which may be incorporated into systems developed for the Government. May be a prime or a subcontractor.
Unlimited Rights Software	RSOs that are developed under a Government contract which are delivered with unlimited rights to the Government.
Restricted Rights Software	RSOs created at private expense that are not in the public domain and which are to be delivered (to the Government) with less than unlimited rights.
Licensed Software	RSOs developed for delivery under Government contract, that the developer retains the right to license for third party use on other Government contracts. This is applicable only if the alternate proposed license approach proposed herein is used.
Software Reuse Library	An element within a software engineering environment used to store and maintain those RSOs that are potentially reusable within and across major programs.
Reuse Library Catalog	A centralized source of information about the RSOs that populate the Software Reuse Library.
Reuse Library Retrieval	Selecting items from a Reuse Library for systems application.
Table 3-1. Operational Definitions of Terms for Software Reuse	

C. LIABILITY

There are two types of legal issues that are pertinent to managing software reuse. They are: the responsibilities with respect to others' intellectual property rights and possible liability resulting from qualifying and distributing reusable components. [Ref. 8:2]

When researching the legal parameters of software reuse it was found more often than not, that managers customarily talked in terms of liability. Liability in reusability seems to be especially a concern within the DoD, having appeared in several of the articles reviewed. In phone conversations, both the Army Intellectual Property Law Division and DISA listed liability as an area of concern, and it is mentioned in the Report of the DoD Ad Hoc Software Strategy Group Meeting. The articles reviewed noted that in today's litigious society, liability must be a concern. [Ref. 9:180] There are three bases of liability: contract, tort and statutory.

1. Liability on the basis of a contract

A contract basis is derived from agreements which might be oral or written, but which better practice says should be written. Essentially, liability in contracts results anytime one party or the other breaches a term (provision) of the contract. For example a subscriber agreement should provide a number of terms and conditions, one of which treats payment (i.e. a subscriber agrees to pay an access charge). Failure to pay access charges as prescribed (amount and frequency) is a breach of the agreement, unless there is some defense which the subscriber could raise (such as excusable delays or waivers).

2. Liability on the basis of tort

A basis in tort exists if there is some legal duty extending from the library to one of the participants (subscriber, supplier or perhaps a third party such as a subcontractor). [Ref. 7:1] There are frequently occasions when duties extend not from the software reuse library, but from the subscriber to the library. Such a duty does not derive from a contract, but from the common law of tort. For example, a library for reusable software components which holds itself out to a community of subscribers to be qualified to conduct testing for conformance of components to standards, such as MIL-STD-1815, and which in fact does so, has a duty to apply that standard of care which pertains to a person assuming such a role. That is, the library, having undertaken to do the testing and

certifying of components for conformance based on the results of that testing, has a duty to adhere to that standard of care appropriate for that role. [Ref. 7:2]

The standard of care required in administering reused software must be defined in terms of whose standard of care the reused software is to be judged by. The question must be asked, is the judgment standard to be that of a registered software engineer or that of a technical librarian. The standard of care determines how well the library must perform its function as a tester and judge as to conformance of the parts.

When the contracting officer advocates the use of software reuse strategies within the acquisition plan, he should expect that contractors will likely refuse liability for the failure of reused software. Therefore the contracting officer should be aware that if the software reuse library from which the contractor obtained the reused code performs its role negligently (breaches its duty to adhere to the appropriate standard i.e. allows substandard code to be stored for reuse), then a basis for a claim in tort might exist, whether or not a contract exists between the supplier or subscriber of components and the library.

3. Statutory Liability

A statutory basis for liability does not require a contract. The Federal copyright statute prescribes violation known as "willful copyright infringement" for which there are provided penalties including fines up to \$100,000 and imprisonment. [Ref. 7:3]

When applying reuse, the different types of liability listed above must be taken into consideration. However, liability concerns can be reduced through the effective use of intellectual property laws which cover data rights, patents, and copyrights.

A U.S. Government employee, acting within the scope of his or her duties, cannot be named as a defendant in a copyright or patent infringement law suit. This immunity extends even to a Government contractor, but only if performing on a contract for the Government. The Government cannot be stopped from using the copyrighted work or

patent idea but will be liable to the owner for compensation for the Government's use of the work or idea.³

D. DATA RIGHTS ISSUES

The majority of those interviewed and the majority of papers reviewed refer to the problem of handling data rights. Data rights are a constant source of confusion. Data rights are hotly debated on a continual basis between system program offices and contractors, each trying to protect what they believe their rights to be. Numerous issues leave many unanswered questions. If the contractor uses software developed at his own expense to develop the program software, does he have to provide his developmental software to the Government? What do restricted rights really mean? [Ref. 1:19]

Data rights have been constantly evolving over the last seven years, and it is still uncertain as to where they will end up. [Ref. 1:19] One view is that software needs to be differentiated from technical data.

The function and purpose of software is different from that of technical data. Software performs tasks; technical data merely conveys information. [Ref. 7:4] Because of this, the economics underlying the development and marketing of software and technical data are significantly different. Software generally involves significant research and development costs which can only be recouped through the marketing of the product, whereas often technical data are generally produced as an ancillary step in the process leading to production of the actual item to be marketed.

The critical point here is that the capital cost of design and development (including the cost of software tools and/or Computer Aided Design/Computer Aided Manufacturing programs which aided in the development effort) are recouped as part of the sale of the system, not through sales of technical data that might have been generated in developing the system. DoD's policy with respect to hardware systems takes this into account by treating hardware systems in a manner different than it treats technical

³ 28 U.S.C. 1498.

documentation. DoD's present policy with respect to software, however, is heavily technical data oriented, and does not allow software design costs to be recovered in the same manner.

Thus, the economics of software development indicate a need for taking software (and the documentation which is an integral part of its development and evolution) out from the quasi-technical data treatment it has thus far received. With regard to development costs and capitalization, software is in many ways more like a hardware component than it is like the technical documentation which supports the hardware. The DoD should consider technical and economic similarities between software and hardware, as well as the dissimilarities between software and technical data. [Ref. 10:3,4]

Differentiating software from technical documentation and associating it more with hardware systems seems to be one of the more important problems with data rights in software. In hardware systems, the Government has rights to the data underlying the development of the hardware device, but continues to purchase the manufactured hardware item on a piece by piece basis. Since software really only exists on paper or magnetic media, it is treated exactly the same as technical data, and the contractor has no ability to sell it to the Government on a piece by piece basis since the Government feels that it has the rights to the product. There is no manufacturing process, other than simple duplication, in replicating software, further blurring the distinction between hardware and software. Little has been done to change this situation, resulting in a continual strain on Government/contractor relations. [Ref. 10:5]

The Federal Government procurement regulations for the acquisition of computer software are based on the Federal Acquisition Regulation (FAR) for civilian agencies and the DoD FAR Supplement (DFARS) for the DoD. The DFARS requires that unlimited data rights for any software developed in whole or in part by Government funds be assigned to the Government. The rationale for unlimited rights is stated in DFARS 227.472-1(a):

For defense purposes, millions of separate equipment and supply items, technical data resulting from research and development and production contracts must be obtained, organized and disseminated to many different users. Finally, the Government must make technical data widely available in the form of contract specifications in the interest of increasing competition, lowering costs, and providing for mobilization by developing and locating alternate sources of supply and manufacture.

Industry, on the other hand, views this problem in a different light. Industry representatives of the Data Technical Working Group (DTWG) reached its own conclusions about unlimited data rights. [Ref. 8:11]

The DTWG found that industry is reluctant to invest in new technology for the benefit of the Government because of sweeping data rights demands by the Government, and apprehensiveness about the loss of proprietary information. This creates a climate unfavorable to the transfer of such technology to the Government.

The Government is failing to obtain the most innovative and creative computer software technology from its software suppliers. Thus, the Government has been unable to take full advantage of the significant American lead in the software technology for the upgrading of its mission-critical computer resources.

Another aspect of this problem is whether or not software should be considered under copyright law or patent law. The problem first developed in 1980 when Congress passed a bill that extended copyright protection to software, making the event the first time that a technology was considered under copyright laws. [Ref. 11:79] Congress compared source code to a book, not realizing that software was instead a technology that could rightfully be covered under patent law. [Ref. 11:79] The next several paragraphs examine the difference between copyright and patent laws.

1. Copyright Law

Copyright law was initially established to protect the expression in a work, and not the ideas in the work. [Ref. 11:80] There are three different types of copyrights: artistic works, factual works, and functional works.

- Artistic works generally enjoy the broadest copyright protection. In general, more elements of artistic works (like a play) will be considered to be expression than in a historical work (like a biography). The law gives artistic works the most protection to encourage artists to seek ever more profuse ways of expressing themselves on themes of enduring interest (the 10,000th novel about love may still teach us something about love, particularly if the law forces it to present it differently than the previous 9,999 novels).
- Factual works have a narrower scope of copyright protection than do artistic works. Factual works fall in the middle because historians, for example, need to be able to draw more on each other's work to advance the progress of knowledge.
- Functional works are afforded a very narrow scope of copyright protection. (Truly functional works, such as chairs or microwave ovens, are because of their functionality not protectable under copyright at all.) More things will be considered to be ideas in functional works (like an engineering drawing) than artistic or factual works. Functional works have the least protection because bridge engineers, for example, have a great need to draw on the work of other engineers who have designed good bridges so they in turn can build safer and more reliable bridges. Furthermore, patent protection is available to reward truly innovative engineering ideas.

Software initially seemed to be like a literary work to Congress because it is written out in source code, just like a draft of a book. Concentrating on this aspect of software obscures its functional aspects and the engineering process by which it is developed. It is more appropriate to think of software as akin to an engineering drawing than to a novel. [Ref. 11:83]

This divergence of opinion between ideas and expression has caused mixed judicial decisions, further clouding a clear understanding of what copyright status exists for software. [Ref. 11:80] Copyright law also works against reuse in the control of derivative works, with the holder of the copyright having the right to control the derivative work. [Ref. 11:85] As stated previously, copyright law is used to protect

interests in software. Patent law also protects interests in software but under different circumstances.

2. Patent Law

Patents derive from a body of law that is designed to protect innovative or non-obvious processes, machines, manufacturing methods, compositions of matter, or any improvements to these. [Ref. 11:81] Patent law forces the inventor to specifically show what is patentable, versus copyright law which "covers the undifferentiated whole of a protected work and unspecified aspects of it that courts may later construe to be an expression". [Ref. 11:85] Patent law also allows, and even encourages, derivative works to further the state of the art, severely limiting the risk of liability to the user. [Ref. 11:85]

It is clear that patents vs copy rights is a confusing legal issue, although it is not clear how it should be best resolved. It is unclear if Congress should modify the past legislation concerning software to remove some of the confusion, or if they should modify both copyright and patent law to handle software more reasonably. [Ref. 12]

The following section describes the effects of the present and proposed FAR and DFARS on software ownership, liability, and incentives to reuse software assets. The focus will be on the business impact of these regulations, as opposed to the legal aspects.

E. THE EFFECTS OF FEDERAL REGULATIONS

Software developers are concerned with a number of regulations which effect them when working for the Government. These regulations:

Give the Government the right, (even if you had developed software at private expense!) to take your software and give it to the Government's support service contractors (perhaps your worst competitor).

Of course the support service contractor is limited in what he can do with that software. But if a competitor gets your source code, it is then in the brains of the competitors' people. From a legal viewpoint, your software no longer has any protection

as "trade secrets", because the knowledge was given them as a result of the Government contract. [Ref. 6:1]

The FAR and DFARS give guidance on how procurement is to be done by and with the Federal Government and DoD respectively.

The regulations affecting right to software are covered in the FAR Section 52.227-14 (Rights in Data) for the non-defense agencies and for defense agencies in DFARS Part 227.4 [Ref. 6:1] and can be applied to common questions concerning software reuse.

1. FAR vs. DFARS

Legal responses to Common Questions of What Software a Contractor Can Reuse based on FAR Part 52.227-14.

a) What software can be reused in the commercial market?

A contractor who develops software under a Government contract is entitled to copyright it for reuse outside the Government market. A subcontractor developing software for a prime contractor under a Government contract can assert the copyright in software (or negotiate with the prime to give him that right).

b) Can it be reused for another contract with a different Government agency?

A contractor can reuse software developed under an Air Force contract or a Navy contract. But keep in mind that the Government cannot pay twice for the same software. For the Navy contract, the Government can only pay for the costs of putting together the software, putting together the documentation, and delivering it. The Navy is then entitled to unlimited rights. A contractor cannot enter into a license agreement with respect to the software delivered to the Navy and cannot charge a royalty (the original developing organization of the reused software has those rights).

c) Are the Government's "unlimited rights" the same as "exclusive rights"?

Case law says that if a contractor develops software under a Government contract, the Governments "unlimited rights" do not mean "exclusive rights." The contractor still has the right to sell the software. However, if the contractor sells it to somebody who is working on a Government contract and charges a royalty, the cost cannot be passed on to the Government.

The present version of the DFARS was published as an Interim Rule in 1988. When published, this interim ruling caused sufficient comment from industry and Government that a new draft "advanced notice of proposed rule making" was published on October 15, 1990. The intent of this proposal is to replace the current DFARS 227.4 (Interim Rule, 1988) and FAR 27.4 with a single regulation addressing rights in technical data and computer software for all Government agencies. Considering the newness of the software field, this appears to have been a reasonable approach at the time. In practice, its shortcomings become clear:

After a period of frustration, it became apparent that it was inappropriate to acquire software as if it were technical data. (The cost of acquiring Government-wide rights-which is what the technical data rights policy provides- to software that was needed at only one Government installation was impeding the acquisition of such software). So software (at least in machine-readable form) eventually became differentiated from technical data in the regulations, although software and technical data policy continue to be somewhat intertwined. Thus, while rights that attach to proprietary software are different from those that attach to technical data, the same standard data rights clause is nonetheless needed to acquire rights in both. [Ref. 13]

Today, the issues are complicated because both the FAR and DFARS have come full circle and treat software as though it were the same as technical data. The FAR and DFARS have basic clauses that address both subjects: FAR 52.227-14, Rights in Data,

and DFARS 252.227-07013, Rights in Technical Data and Computer Software. This is true despite the fact that technical data and computer software are very different [Ref. 5].

In Table 3-2, notice how definitions of computer software and data (in the first two rows) are different for the FAR and the DFARS. In the third column, the proposed change will continue the combined treatment of data and software. As long as this is the case the issues will continue to be confused.

Terminology	Federal Acquisition Regulation	Defense Federal Acquisition Regulation Supplement	Proposed Regulatory Change (October 1990)*
Software	Computer programs, computer databases, and documentation thereof (FAR 27.401)	Computer software and computer databases (DFARS 227.471). Treats documentation as technical data, not software.	Combined treatment of data and software.[Ref 5]
Data	Recorded information, regardless of form or the media on which it may be recorded. Term includes technical data and computer software (FAR27.401).	Recorded information, regardless of form or the media on which it may be recorded (DFARS 252.227.7013).	-
Copyright	Contractor authorized to establish copyright claim. Government granted paid-up, nonexclusive, irrevocable worldwide license to reproduce, prepare derivative works,	Contractor authorized to copyright unless software is considered a "special work," when software becomes sole property of Government and is treated as unlimited	More favorable than before. Does not include Government right to distribute copies, as now found in DFARS, but does allow full disclosure for Government

	perform, or display publicly by or on behalf of the Government. License does not include right to distribute software to public (FAR 27.404(f)(iv)).	rights software (DFARS 227.476/252.227-7020). Government granted a nonexclusive, paid-up license to reproduce, to distribute to the public, to perform or display publicly, and to prepare derivative works and have others do for Government purposes (DFARS 227-7013(3) 227.480/252.227))	purposes. No resolution of derivative works issues.
Government purpose License Rights (GPLR)	No definition.	Right to use, duplicate, or disclose data and software, in any manner for Government purposes. This includes competitive procurement but not commercial purposes. Government can authorize others to use for Government purposes.	Newly applies to software; contractor retains exclusive commercial rights for a negotiated period of time. Allows negotiation in cases of mixed funding by developer or Government.
Table 3-2 Comparison of FAR, DFARS, and Proposed Regulatory Change ⁴			

This thesis only scratches the surface of the complexity in the many buyer-seller relationships found in the Government software arena. These include relationships among owners of reusable software assets, developers, third-party developers, and customers (see Table 3-1 for explanation of terms). Figure 3-1 illustrates the number and

⁴ STARS (1991): FAR (1988), DFARS (1990).

variety of sources for software assets that are reusable to a greater or lesser degree. The single largest collection of existing software is found in the DoD inventory (lightly shaded in the figure), some of which can be traced back three decades. A second category is public domain software (unshaded). The third category of reusable software assets (more heavily shaded) represents organizations with the mission to develop software for sale. Organizations in this category are now seriously affected by the Government's regulations and procedures for acquiring reusable software. These organizations' concerns include questions of ownership, liability, and incentives to both create reusable software and to reuse existing software.

The results of interviews conducted with DoD software personnel addressing these concerns are presented in the next chapter.

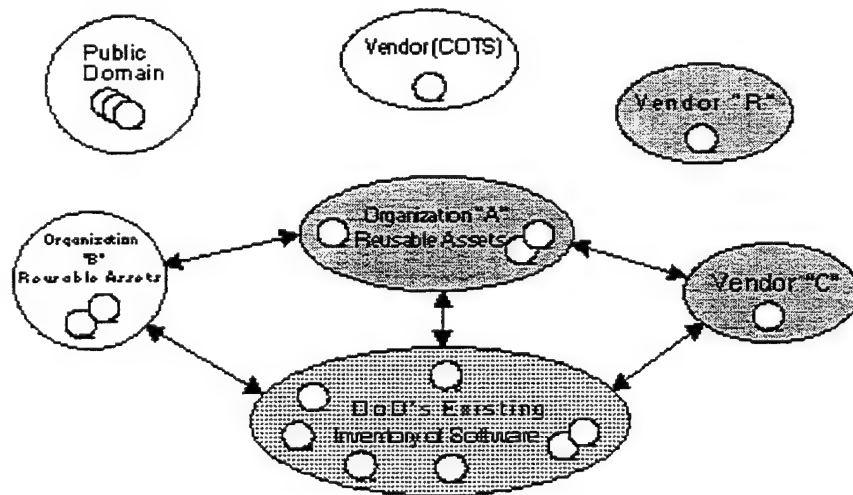


Figure 3.1. Software Sources.

IV. METHODOLOGY AND ANALYSIS

A. METHODOLOGY

1. Methods

This thesis relies on evidence gathered through literature review, telephone interviews, and electronic mail queries to identify what legal issues are prevalent in the software reuse industry and what is currently being done to control and clarify those issues. Those interviewed consisted of legal officers from the U.S. Army Intellectual Property Law Division, Software Development Specialists from the Defense Information Systems Agency (DISA), and Software Procurement Specialists from the Marine Corps Tactical Support Systems Activity.

There are two steps in completing the evidence gathering for this thesis. First of all, a series of open-ended telephone interviews were conducted to determine what intellectual property issues confronted the software reuse industry, more specifically the area of patents, data rights, and copyrights. These interviews also attempted to determine what Government and DoD personnel encounter when working with developers on intellectual property issues. These questions were put forward to gather opinions on the subject of software reuse and patents. The response was that patents on software were going to monopolize the basic concepts involved in software reuse and that encouraging the expansion of software patents was not a good idea.

Based on this initial question, a revised question was posed and a more structured response was requested. The interviews were conducted with the researcher posing the basic question and then following up with additional questions on an open-ended basis based on the response. The responses were then compared and collated into one response representing the majority of answers received.

2. Justification of Methodology Choice

The method of choice for research depends on (1) the degree to which the investigator can control the situation, and (2) whether the emphasis is on contemporary versus historical issues. [Ref. 14:16,17] Within these three conditions are perspectives that must be considered in determining the research design:

- a. The degree to which the research problem has been crystallized (the study may be either exploratory or formal).
- b. The method of data collection (studies may be observational or survey).
- c. The power of the researcher to affect the variables under study (the two major types of research are the experimental and the ex post facto).
- d. The purpose of the study (research studies may be descriptive or causal).
- e. The time dimension (research studies may be cross-sectional or longitudinal).
- f. The topical scope, breadth, and depth of the study (a case or statistical study).
- g. The research environment (most business research is conducted in a field setting, although laboratory research is not unusual; simulation is another category, somewhat similar to laboratory research). [Ref. 15:59]

3. Methodology Perspectives

a) Exploratory versus Formal

The difference between these two types of studies pertains to the degree of structure and the objective of the study. Exploratory studies are rather loosely structured with the objective of developing hypotheses for future studies. Formal studies use a more structured approach and take the hypotheses generated by an exploratory study and attempt to prove them. [Ref. 15:60]

The first step of this thesis was exploratory in nature, with the outcome being a set of guidelines to look for when implementing software reuse programs.

b) Observation versus Survey

In observation studies, the researcher observes some situation without asking questions of the people involved in the situation. Surveys allow the investigator to probe people for their responses to questions. Within the context of surveys are formal surveys and interviews that are either conducted through the mail, electronic mail, over the phone, or in person. [Ref. 15:60]

A combination of observation and survey was thought to be the best methodology to determine the current state of software reuse legal issues. Initial research indicated that there was a very limited list of programs that had attempted, or are currently implementing reuse. In addition, it became apparent that reuse legal issues were very sporadic and *ad hoc*, so much so that a formal survey was determined not to be appropriate at this time. Based on these problems, it was decided that telephone interviews, and electronic mail queries were the most appropriate instrument for gathering information.

4. Data Analysis Methodology

The responses to the structured interviews were analyzed by determining appropriate categories of responses for each question asked, tallying the responses in each category, and then comparing the tallied responses with actual application and analyzing the similarities and differences. Most of the responses to the questions lend themselves to opinion.

B. ANALYSIS

The term software reuse in its basic form was found to be highly technical. To apply it to actual activities however requires a whole other set of deliberations, i.e. that of major legal considerations. Specifically the major concerns are that of intellectual property rights i.e. data rights, patents, licenses, and ownership. The FAR and DFARS clearly are the guiding tools which govern how contracts are to be patterned and interpreted when presiding over software reuse issues. This guidance in the software

industry as a whole, not just software reuse, is lengthy, complicated, and consists of much duplicity. The majority of legal software reuse issues such as data rights, and patents, are no different than original software issues.

The analysis of the findings focuses on the areas discussed in the previous chapter and then proceeds to propose explanations on why the DoD has run into so many legal impediments to their software reuse program. Finally, the research and investigative questions will be addressed, and conclusions will be drawn based on the hypothesis analysis.

The personnel selected represented a cross-section of individuals who are or have been involved with reuse in the DoD community. Each question from the interviews is presented along with the responses received. An aggregate of the responses follows each question. No statistical analysis is performed using these data. Quotations are provided on a non-attributational basis due to the majority of people requesting anonymity. The interpretations of the results follow each response respectively.

C. RESPONSES FROM PHONE CALLS

Question 1. How does the software reuse industry view software patents?

The responses range from "software patents will have an extreme effect" to "they will have somewhat of an effect" on the creativity which goes into software development. One hundred percent of the respondents indicated that there are no clear criteria to distinguish between software which is patentable and that which is not. Two of the respondents felt that protection is necessary for those who build software from scratch.

Analysis. Many concerns have been expressed recently by the computer industry about software patents. Programmers have a legitimate fear that patents will be granted that monopolize the basic concepts that are involved in user interfaces or the algorithms that programmers need to write effective software. It has been claimed that the U.S. Patent and Trademark Office seems to be willing to issue virtually any software patent presented to it, that it lacks the facilities to search out prior art in the field, and that the

personnel available are so limited that the Patent and Trademark Office is prevented from effectively examining patent applications. Patent applications stay on file in the Patent and Trademark Office in secret for an average of about three years before they are issued and published. [Ref. 7] The results are businesses may invest considerable resources to develop and establish a product that suddenly becomes infringing when a patent springs into existence. The U.S. patent system, unlike those of many other countries, provides a more or less absolute monopoly for the patent holder. A patent holder can charge as high a price for a license as it chooses and can refuse licensing altogether. Yet, software algorithms, just as much as any other advances, are deserving of the incentives and rewards of the patent system.

Software patent exposure is still relatively new to most industries, however patent attorneys view software patents as a potential bombshell with respect to cutting down incentives for the development of software technology. This is mainly because there are no clear criteria for discriminating between software that is patentable and software that is not. There are not enough court decisions on record as of yet to establish a pattern.

When software was used in the process of vulcanizing rubber⁵, the Patent and Trademark Office has said, software is now patentable. There is great worry about what software patents will do to the inventiveness that is involved in software. Small businesses cannot afford the legal fees for patent applications, enforcing one's patents when necessary, and defending against patent infringement lawsuits filed by others. Even for large businesses, patents for software can become a legal nightmare, because companies can make huge investments, and subsequently find out their product has infringed upon another's patent rights.

There is major concern on how to protect the person who has invested money in building software from scratch. Software patents may stifle creativity at some point. If someone is investing money in building a product, there has to be some protection that

⁵ To treat (rubber) with sulfur and heat, thereby imparting greater strength, elasticity, and durability.

allows him or her to get a return on that instrument regardless of whether it is being used as original software or in part as reused software.

There will in the future be a great deal of litigation that is going to arise in the area of software and software reuse patents and, especially with software reuse all the complications that are involved with third and fourth parties. It's going to be a great challenge for companies large and small to navigate the software patent maze.

Question 2. What rights should software technology inventors have with respect to their inventions?

None of the respondents felt that software technology inventors should give up all their technical data rights to the Government. However, the majority of the respondents indicated that if compensated fairly inventors would be willing to participate in most software reuse programs.

Analysis. The history of software protection is based on many cases since the early 1980's when it was argued that computer programs should not be protected by copyright, in part because it would give too great a monopoly to computer programmers. Another argument that was made was that the patent system was a more appropriate system for the protection of software. According to this argument, software is primarily concerned with algorithms and processes that are implemented, these are ideas that are applied in a certain way in a technological device, and that this was therefore a proper subject matter for patent protections. The patent system had safeguards--a way of assessing prior art, a higher standard for giving protection, and a shorter term of protection than the copyright. The fundamental issue for copyright protection of software from the developer's perspective that is now being confronted in the courts is just how far section 102(b) of the U.S. Copyright Act will extend. That is the section that exempts ideas and methods from copyright protection processes,

Question 3. How would you improve the current patent system as it pertains to software reuse?

Only a small number of the respondents felt qualified to answer this question. The response was that the patent process for software was "entirely too long", and by the time a patent was received much of the technology had already been compromised to competitors.

Analysis. A common consensus is to reduce the current life of patents from 17 years to a shorter period. The patent lifetime should also begin at the date of application rather than the date of issue. Some patents spend so much time in the application process that their life nearly doubles.⁶

Requiring the mandatory licensing of patents is another idea, while still another is to restrict the rights of patent holders that are not making use of the patent. Finally, there is a need to restrict the abuses of the patent system by companies that buy patents and then seek license fees from other companies.

Government procurement regulations are different from other systems for allocating intellectual property rights and responsibilities in that they are written by the consumer. As a result, some of the procurement regulations have seemed unbalanced in favor of the Government as contrasted with more traditional areas of intellectual property law which tend to favor the creators of works. [Ref. 7:7] Emphasizing the need to provide incentives for creative, skilled people to motivate them to continue to produce items that would benefit society, the software industry has claimed that broad claims of rights by the Government inhibit their ability to commercialize software technology and thereby recoup their investment. An example of this is industry's concern that if they permit the Government to have access to valuable source code and other technical data

⁶ An example of a patent that took a long time in the application process is one issued in July 1990 to Gilbert Hyatt covering the invention of the microprocessor (Single chip integrated circuit computer architecture, U.S. Patent 4,942,516). The Inventor's first application was received in 1970. However, in Hyatt's case, even if his patent is successfully defended he will get royalties only for seventeen years after the patent was issued, since he was not involved in the commercial development of the microprocessor and has as yet received no royalties. Some developers can keep their invention a trade secret while in the application stage, thus effectively lengthening the life of the patent.

containing proprietary information for reuse, this material may end up in the hands of their competitors.

Question 4. What impact on reuse does industry's concern that their proprietary data will fall into the hands of their competitors, have when contracting with the Government?

The respondents indicated when dealing with industry their main concern was that the Government is known for requesting full rights to technical data, and that software reuse would only reduce the amount of royalties they would receive. One of the respondents mentioned that reuse already goes on within the company, problems only arise when the Government wants to allow another contractor to use their software.

Analysis. Reuse has been addressed from two perspectives: internal reuse and external reuse. Internal reuse, in which the company reuses components internally is happening today and there are no major problems regarding intellectual property rights. External reuse, though, will have to occur if we are going to create a reuse industry, and in order to make this happen the Government is going to have to allow industry to maintain rights to the intellectual property that they have created to prevent compromising their intellectual property rights to their competitors.

One of the big problems is industry's tendency to identify its proprietary rights by designating the data as limited rights data when it delivers software to the Government, and third parties that are not Government contractors are precluded from getting that data. If we are going to have a reuse library that is going to be effective, there should be a level in that library where the development of derivative software by third parties does not require any passage of royalties or incentives back to the contractor that developed the original software. Otherwise you will have an administrative nightmare. The results would be a succession of derivatives where it would be impossible to determine at what point you were in this line of progression. A suggestion is that after the second derivative contractor any reimbursement to the developer would stop. Beyond the second derivative, the administrative cost would be likely to exceed the remuneration to the

developer. As for Government infringement on the proprietary rights of the software developed by contractors, that depends very much on the particular agency involved. In general, Government regulations pertaining to data rights are not necessarily viewed as too restrictive however, they are viewed as too confusing. The FAR regulations are viewed as good, yet the DFARS are viewed as lousy. About 90% of DFARS data rights regulations are viewed as being redundant. [Ref. 7:8]

Unfortunately, the final answer to any dispute over rights to reuse a component ultimately becomes avoiding reuse and developing the code internally. The economic realities will drive the decisions that are made-if a prime contractor is obligated to deliver something and he cannot get the rights that he is promised from a subcontractor then he will in all probability make arrangements to program it himself. While this defeats the purpose of reuse, these are still the economic realities under which we will be operating for at least the next few years.

V. CONCLUSION AND RECOMMENDATIONS

There are many issues which are critical to effective software reuse, however, as highlighted throughout this thesis there are some core legal impediments to software reuse. This thesis has attempted to enlighten the reader on legal issues that must be resolved in order to establish a successful reuse industry.

A. THE MAJOR CONCERNS

There are many unknowns with respect to interpretations of software reuse in the FAR and DFARS. Contractors have concerns about ownership rights which raise questions about the extent of their intellectual property rights (i.e., copyright and patents). Also, liability and licenses agreements need to be established or addressed.

Access to and protection of proprietary software are issues which can potentially inhibit large scale software reuse. For software reuse to be successful, public access to reusable assets is essential. Proprietary software issues must be addressed that would allow this public access but also protect the investment by the developers.

There were several conclusions reached as a result of this thesis, however it is obvious that the practice of software reuse is evolving even with the myriad of stumbling blocks placed before those who wish to advance its practice. Large-scale reuse will probably require a registry that allows tracking the source of original development and modifications for each component. The establishment of a national registry for reusable software components is needed in order to track the extent a particular string of software is reused. Developers would like to see the Government assume all legal liabilities when the contractor is required to implement reused software in its development plan. This practice would probably conflict with federal law.

It is not likely that the uncertainties involved with intellectual property protection for software and liability for software malfunction will be resolved soon. In the realm of intellectual property rights, software reuse raises few, if any, fundamentally new legal

issues that have not been confronted before, but large-scale reuse will likely result in these issues being encountered more frequently and in more complex ways.

In order for a large-scale reusable components industry to develop, it is beneficial for mechanisms to be implemented that can appropriately reward developers who modify and add value to existing components, while still protecting the rights of the developers of the original components. Developers feel that patents are a viable way of protecting their ideas. Extensive research has been done in the area of patenting software. Software patents may have significant potential as a mechanism for encouraging the development of reusable components, however, there are many uncertainties about the validity of software patents and the impact of patents and patent infringement lawsuits on the software industry that need resolution.

Licenses for reusable components have become the means by which developers are recouping royalties for the authorized use of their developments. Licenses however are not yet completely irrevocable, do not yet allow liberal rights to produce derivative works, do not yet require the developer to fix errors, and do not yet contain provisions for determining who is allowed access to components.

B. INHERENT PROBLEMS

A report by The National Security Industrial Association (NSIA) urges that the Government continue to support initiatives in the area of software data rights

No data rights issues emerged that were uniquely associated with reusable software (emphasis added). Data rights issues associated with software reuse are very important and must be resolved. However, it turns out that the set of problems associated with rights to reused software are precisely those that have been and are currently being addressed by Government and industry. (NSIA 1990,21)

The Government should continue to vigorously support on-going initiatives in the area of software data rights. Related to data rights, nothing additional needs to be done

by the Government in order to promote the concept of reuse, other than to support the software data rights initiatives already in progress. (NSIA 1990,21)

Liability issues will continue to plague the software reuse industry until solid mechanisms are put into place which will identify who bears the liability. It is in the best interest of the Government not to claim all liability associated with software reuse. This type of situation would foster an atmosphere of non-responsibility on the part of developers. The risks of liability due to reuse of software are important and deserve serious management attention.

Once software is modified by someone other than its owner, the owner's relation to that software becomes almost zero. In developing DoD weapon systems, software being reused will almost certainly require some modification. Consequently, the system developer will bear the primary liability for the modified software. The many liabilities of ownership, especially legal are perceived deterrents to the voluntary engagement in software reuse.

C. RECOMMENDATIONS

1. Education

There is a great need for all involved in software reuse to be fully aware of the legal pitfalls when reusing software. Lawyers should not and cannot be the only source of resident knowledge to the proprietary rights of software.

The DoD is currently making strides in ensuring its acquisition force is competent and has a good source of dedicated individuals. However, these individuals need to be adequately trained for the role they have to perform in complex software acquisition contracts. Much of the software that the contracting personnel must acquire is "state of the art" technology. [Ref. 7:11] Communication between procurement personnel and users must be more frequent in order to make maintenance and support less difficult. Procurement personnel need to have more training in software technology, software life cycles, and software support systems. Additionally the procurement

regulatory structure within which the negotiation process must proceed, especially as to data rights, is quite complex. Therefore, more time must be given to procurement personnel to learn the complex issues involved with inherent concerns such as data rights.

2. Acquisition Professional Recommendations

The first thing acquisition specialist should do is specifically define contractual wording for reuse incentives so that developers know from the outset that they will be using reused software or that they will be designing software to be reused. Next, contractual language must be written for liability avoidance. That the Government will not be liable for the failure of reused software must be clear from the very beginning of the acquisition process. Last, specific evaluation factors for software reuse must be developed in order to adequately decide where the best software plan is coming from.

Once the contract has been written, contract evaluators must be made clear on understanding the software reuse part of the evaluation. There are inherent increased costs involved in developing software to be reused and there are inherent decreased costs involved once reused software is being used. Next, program managers must monitor the metrics of software reuse on their projects in order to mitigate software risks.

There are specific actions which developers can take prior to and during the development process to ensure that their rights are protected. In the case of privately funded software the following are recognized as viable solutions:

- a. Have in place a mechanism (system, procedure, discipline) for ensuring that it can prove that it develops its software at private expense. Before entering into a contract, a contractor must specify in an agreement which computer software developed with the contractor's funds it intends to supply to a DoD agency or a prime contractor with restricted rights, not unlimited rights. This agreement must be incorporated into the contract. The company then represents that it is providing a correct written assertion of restricted rights data and that the company is entitled to limit the Government's rights.

A contractor must be able to prove that it develops its software at private expense before any Government contract. Thus, the person who signs the certificate must

know or have a reasonable basis to believe that he can restrict the Government's rights in this matter. He must convince the Government that the software was (a) developed before any Government contract and (b) was developed without any Government funding. [Ref. 7:8]

Companies need internal procedures to prove their assertions. They must be able to document the stages of development of their privately funded software. To face a challenge by the Government, the company must be able to prove that the software was workable (to a degree that people skilled in the art would say "that software will work") before it signed that contract.

b. When dealing with an executive agency other than the DoD under the new regulations, state in the agreement the characteristics of the privately developed software that will be recognizable if the Government should choose to modify it. [Ref. 7:9]

c. Do the best it can to avoid delivering privately developed software to the Government. If the Government requires delivery, the company can try to negotiate constraints on what the Government can do with it. A subcontractor, dealing with a prime contractor, will have somewhat more leverage when it attempts to negotiate this. [Ref. 7:10]

Companies must be able to show the Government that the software they developed was at their own expense if they want to prevent the Government from infringing any data rights. Until it can show that, it cannot restrict the Government's rights in any privately developed software. Current regulations make it difficult for companies to make their cases when it comes to limiting the Government's rights to software.

The area of software reuse and its associated legal issues is an extremely complex and controversial topic. Intellectual property rights mechanisms, such as copyrights, patents, and trade secrets, often pose substantial barriers to reuse. These mechanisms need to be studied so that ways of eliminating or reducing these barriers can be found, and the mechanisms harnessed appropriately so as to help provide economic incentives for reuse. [Ref. 13:39]

D. AREAS FOR FURTHER RESEARCH

There are many issues which continue to slow the software reuse process, some are legal issues, some are incentivising, and yet still some are economic benefit issues. All these areas have remaining problems which need to be clarified and or solved. The following areas offer opportunities for further study and will necessarily need to be addressed in the course of DoD software reuse implementation.

1. Examination of current proposals to the FAR and DFARS which address software reuse as it pertains to intellectual property, and then compare past practices to see what areas of existing software reuse plans will be impacted.
2. Examine the potential impact of a software reuse library for use by civilian and DoD agencies to limit the cost associated with maintaining a software reuse depository.
3. Explore what can be done to encourage companies to create software that is reusable, both by themselves and, eventually, by other firms.
4. Explore what encouragement would result in the utilization of software previously developed for reuse.
5. Examine the fiscal barriers to software reuse and determine if the additional inherent cost of developing software outweigh the inherent savings when using reused software.
6. Explore the liability of software developers resulting from the malfunction of software as it pertains to who is responsible when software, that is originally developed by one organization and reused by another, malfunctions.

These areas of further study are not an all inclusive list of the remaining impediments to software reuse, however they are a good indicator of the non technical barriers which continue to hamper the software reuse industry. While few question the economic benefits of reusing software, there is much to be said for developing a software reuse program, most prominently the cost involved in developing and maintaining a

viable program. History shows that the ability to reuse an object eventually saves on the cost of developing and using that object.

LIST OF REFERENCES

1. Software Reusability: A Study of Why Software Reuse Has Not Developed Into a Viable Practice in the Department of Defense, Thesis, Holmgren, Brian, Captain, USAF.
2. *Software Reuse Guidelines*, ASQB-GI-90-015. Atlanta, Georgia: Georgia Institute for Technology.
3. *Software Reuse Survey Report*. Sterling, Virginia: Software Engineering Guild.
4. Software Technology for Adaptable, Reliable Systems (STARS) 1991, Reusable Software Acquisition: Current FAR and Budget/Finance Requirements, GR-7670-1226(NP). Reston, Virginia: Unisys Defense Systems.
5. Samuelson, Pamela. "Is Copyright Law Steering the Right Course?" IEEE Software, 78-86 (September 1988).
6. DeVecchio, W. Jay. Legal Barriers to Reuse and Liability, SPC-90093-N. Video presentation to the Software Productivity Consortium. Herndon, Virginia: Software Productivity Consortium.
7. Abstraction-Based Reuse Repositories, REUSE_REPOSITORIES-89041N. Herndon, Virginia: Software Productivity Consortium.
8. Probert, T. (1984). Report of the rights in Data Technical Working Group (RDTWG), Volume 1: Executive summary (pp. 88-92). Alexandria, VA: Institute for Defense Analysis. IDA Document D-754.
9. Bott, M.F. and P.J.L. Wallis. "Ada and Software Reuse, " Software Engineering Journal 3,5: 177-183 (September 1988).
10. Proposal for a New "Rights in Software" Clause for Software Acquisitions by the Department of Defense. Technical Report SEI-86-TR-2, Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA 15213, September 1986.
11. Peterson, A. S. (1989). Coming to Terms with Terminology for Software Reuse. In J. Baldo (Ed.), Reuse in practice workshop summary (pp. 88-92). Alexandria, VA: Institute for Defense Analysis. IDA Document D-754.
12. Draft Army Software Reuse Study, provided by Julie Allen of SAIC. January 1990.

13. Samuelson, P., Toward a Reform of the Defense Department Software Acquisition Policy, CMU/SEI-86-TR-1. Pittsburgh, Pennsylvania: Software Engineering Institute.
14. Strategy and Mechanisms for Encouraging Reuse in the Acquisition of Strategic Defense Initiative Software. IDA Paper P-2494, by James Baldo and Craig A. Will. Prepared for Strategic Defense Initiative Organization. Institute for Defense Analyses. (June 1990).
15. Emory, C. William Business Research Methods. Homewood IL: Richard D. Irwin, Inc., 1985.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
 Cameron Station
 Alexandria, VA 22304 - 6145

2. Library, Code 52.....2
 Naval Postgraduate School
 Monterey CA 93943 - 5101

3. William Short, SM/Sh2
 Naval Postgraduate School
 Monterey CA 93943 - 5101

4. Mark Stone, SM/St2
 Naval Postgraduate School
 Monterey CA 93943 - 5101

5. David V. Lamm, SM/Lt.....2
 Naval Postgraduate School
 Monterey CA 93943 - 5101

6. Claxton R. Johnson, Jr.....2
 3338 North Wilson Road
 Oak Harbor, WA 98277

7. Defense Logistics Studies Information Exchange.....2
 United States Army Logistics Management Center
 Fort Lee, VA 23801-6043